Evolutionary Interactive Bot for the FPS Unreal Tournament 2004

José L. Jiménez[1], Antonio M. Mora[2], Antonio J. Fernández-Leiva[1]

[1] Departamento de Lenguajes y Ciencias de la Computación,
Universidad de Málaga (Spain).
josejl1987@gmail.com,afdez@lcc.uma.es
[2] Departamento de Teoría de la Señal, Telemática y Comunicaciones.
ETSIIT-CITIC, Universidad de Granada (Spain).
amorag@geneura.ugr.es

Abstract. This paper presents an interactive genetic algorithm for generating a human-like autonomous player (bot) for the game Unreal Tournament 2004. It is based on a bot modelled from the knowledge of an expert human player. The algorithm provides two types of interaction: by an expert in the game and by an expert in the algorithm. Each one affects different aspects of the evolution, directing it towards better results regarding the agent's humanness (objective of this work). It has been conducted an analysis of the experts' influence on the performance, showing much better results after these interactions that the non-interactive version. The best bot were submitted to the BotPrize 2014 competition (which seeks for the best human-like bot), getting the second position.

1 Introduction

Unreal Tournament 2004[1], also know as UT2004 or UT2K4, is a first person shooter (FPS) game mainly focused on the multiplayer experience, which includes several game modes for promoting this playability. It includes a one-player mode in which the rest of team partners (if so) and rivals are controlled by the machine, thus they are autonomous. They are called *bots*.

One of the main contributions of Unreal series is the utilities that the game offers for the gamers to create and share their own creations or modifications on the game contents (*mods*). These could be new weapons, scenarios, and even new game modes. Most of them are, in turn, more popular among the players than the original ones. They can be created with the editor that the game provides (UnrealED), included with every copy.

The Artifical Intelligence (AI) has completely changed in the last years, from the classical set of static behavioural rules (including some random component), the use of Finite State Machines (FSM), or the recent use of Scripts, Navigation Meshes and Behavioural trees. The most extended among the *Non-Player Characters* (NPC) could be the FSMs [2].

However, nowadays the AI is focused, not only in defining very competitive rivals or partners, but also in the humanness of them. This aims to design 'be-

lievable' NPCs to which the player can feel empathy, in order to improve his/her experience and lasting appeal about the game.

This paper presents a work in this line, proposing an Interactive Genetic Algorithm (IGA) [3] which evolves a bot, previously defined, by modelling the knowledge and tricks of an expert human player, named Expert UnrealBot [4]. The objective is that the bot has a human-like behaviour, rather than just a very competitive bot with a very good performance in the game.

This bot has been designed to play in the *Deathmatch mode*, in which the aim in a match is to reach a number of *frags* (enemies killed) or the maximum amount of frags in a limited amount of time. There could be just 1 vs 1 player or several players fighting against others.

The presented approach includes two types of interaction: the first one is conducted by an expert in the game (expert human player in UT2K4), who can assess the behaviour of the bot considering its humanness; the second type of interaction is done by an expert in the algorithmic part, i.e., he/she can appraise the performance on an individual in the algorithm.

We have also defined a parallel approach which can profit the run in a cluster of computers in order to reduce by ten the computational time (which was very expensive).

2 Background and related works

2.1 Turing Test for bots

This is a variation of the classical Turing Test in which a human judge who looks and interacts with a virtual world (a game), must to distinguish if the other actors (players) in the game are humans or bots. This test was proposed in order to advance in the fields of Artificial and Computational Intelligence in videogames, since a bot which can pass this test could be considered as excellent, and could increase the quality of the game from the players' point of view. Moreover the test tries to prove that the problem of AI for videogames is far from being solved. Aiming to other areas, the methods used in this test could be useful in virtual learning environments and in the improvement of the interaction human-robot (or machines in general).

The Turing Test for bots is focused on a mutiplayer gamer in which the bot has to cooperate of fight against other players (humans or bots), making the same decisions that a human would take. This was transformed into an international competition with the features:

- There will be (ideally) three players: a human player, a bot and a judge.
- The bot must 'simulate' to be more human than the human player, however the marks are not complementary, i.e. both players can obtain a mark between 1 and 5 (1=Non-human, 5=Human).
- The three players cannot be distinguished from 'outside' (even with a random name and appearance). Thus, the judge cannot be influenced by these features.

– There is no chat during the match.
– Bots cannot have omniscient powers as in other games. They can just react to the same stimuli (caught by means of sensors) than a human player. The human player must be an averaged one, in order to avoid very competitive behaviours that could also influence in the judge's decision.

In 2008 it was held the first 2K BotPrize competition (BotPrize from now on), in which UT2K4 was considered as the 'world' for this test. The participants should create their human-like bots using an external library to interact with the game by means of TCP connections, named GameBots [5].

The first BotPrize competition considered the Deathmatch mode, and divided the rounds in 10 minutes matches. The judges were the last joining to the matches and observed on player/bot exactly once and none bot was categorised as more human than a human. In the following editions (in 2009, 2010 and 2011) the marks of the bots were improved, however they still were not able to overcome to any of the human players. Anyway, the maximum humanness score for the human players was just 41.4 %. This demonstrates the limitations of the test (or the competition), since even appraise a human behaviour is a quite complex task.

2.2 Human-like bots

The winners of BotPrize 2012 were *UT*2Bot [6] and *MirrorBot* [7]. The first one was created by a team of the University of Texas. This bot is based in two main ideas: the application of multiobjective neural evolution in order to use just those combat actions which seem to be human-conducted and a proper navigation of the map. They used stored logs of human plays in order to optimize both objectives. MirrorBot was created by Mihai Polceanu (a PhD student in the LAB-STICC, ENIB). It basically imitates the reactions of the opponent, using interactive techniques. The idea is that if a judge is close to the bot it would imitate his/her behaviour, which could clearly mislead the judge and make him/her thinking that the player is showing a human-like behaviour. Obviously this technique is not useful if the opponent is another bot.

The bot we are presenting here is named *NizorBot*, and it is an improved version of our previous ExpertBot [4]. The aim of the latter was the modelling of the behaviour of an expert Spanish human player (one of the authors), who participated in international UT2K4 contests. It included a two-level FSM, in which the main states define the high level behaviour of the bot (such as attack o retreat), meanwhile the secondary states (or substates) can modify this behaviour in order to meet immediate or necessary objectives (such as taking health packages or a powerful weapon that is close to the bot's position). The decisions about the states is made by an expert system, based in a huge amount of rules considering almost all the conditions (and tricks) that a human expert would do. It also uses a memory (a database) to retain important information about the fighting arena, such as the position of items and weapons, or advantageous points. This information is stored once the bot has discovered them, as a human player would do.

ExpertBot is formed by two layers: The first one is the *cognitive layer*, in charge of controlling the FSM considering the environmental stimuli (perceived by sensors). It decides the transitions between states and substates using the expert system and the memory (database). The second one is the *reactive layer*, which does not perform any kind of reasoning, and just react immediately to events during the match.

3 Methodology

Human intervention for guiding the optimisation process has proved to be very effective improving, not only the quality of the solution, but also the performance of the algorithm itself for finding the solution to a problem [8]. This improvement is specially needed for problems where the subjectivity is part of the evaluation process. This is typical of problems which involve human creativity (such as generative art), or those which aims to increase the human satisfaction or challenge (such as content generation in games).

However, the interaction in an optimisation algorithm could be conducted in many different ways and could affect to many different aspects of the approach. On the one hand, the participation of an expert in problem scope would be key in order to measure the quality of a candidate solution. On the other hand, an expert in the algorithmic scope could improve the overall performance of the approach or could direct the search/optimization process to promising areas of the space of solutions [9].

Even though these methods perform very well, interactive algorithms are not very common since they have some flaws, including the human tiredness or boringness if the expert has to intervene several times (which should be, in turn, ideally).

The usual solution [10] consists in adapting the algorithm to be more 'proactive', so it could predict somehow the decisions that the human will take about a candidate solution. Another approach is based in the 'extrapolation' of the decisions taken by the human, so similar solutions (closer in Euclidean distance for instance) would receive automatically a similar value. However this approach depends on the problem definition and on the structure of the solutions, since close solutions in the space of solutions should be really similar in the problem scope.

In this work, we propose an Interactive Genetic Algorithm (IGA)[3], in which the experts guide the optimization of the ExpertBot parameters in order to obtain a human-like bot. We think that this intervention is very relevant in order to value the candidate solutions (i.e. bots) regarding something as subjective as the humanness of an autonomous agent in a game.

In the following sections the algorithm and the points of interaction are described.

3.1 Chromosome structure

Every individual in the GA is a chromosome with 26 genes, divided in 6 blocks of information. The description of each block is:

- *Distance selection block*. Set of parameters which control the distance ranges that the agent considers to attack with one specific type of weapon, and also the distances to the enemy both for attacking it or for defending from it.
  - *Gene 0 :* Short distance. Value between 0 and 1200.
  - *Gene 1 :* Medium distance. Value between the current value for Gene 0 and 2000.
  - *Gene 2 :* Far distance. Value between the current value for Gene 1 and 2800.
- *Weapon selection block*. Set of parameters which control the priority assigned to every weapon considering some factors, such as distance to enemy, altitude where the enemy is, and the physical location of the agent with respect to the enemy.
  - *Genes 3-11:* They control the priority associated to every weapon. Values between 0 and 100.
- *Health control block*. Set of parameters which control the level of health of the bot, depending on which it will be offencive, defencive or neutral.
  - *Gene 12:* If the health level is lower than this, the bot will be defencive. Value between 0 and 100.
  - *Gene 13:* If the health level is lower than this, the bot will be neutral. Value between the current value for Gene 12 and 160.
- *Risk block*. Set of parameters which control the amount of health points that the bot could risk.
  - *Gene 14:* Value between 5 and 30.
  - *Gene 15:* Value between 15 and 80.
  - *Gene 16:* Value between 15 and 60.
  - *Gene 17:* Value between 10 and 120.
  - *Gene 18:* Value between 20 and 100.
- *Time block*. Gene which defines the amount of time which the agent considers to decide that the enemy is out of vision/perception.
  - *Gene 19:* Value between 3 and 9 seconds.
- *Item control block*. Set of parameters which control the priority assigned to the most important items and weapons. As higher the value is, higher the priority will be for 'timing' the item/weapon (i.e. control of its respawn time).
  - *Gen 20:* Priority of Super Shield Pack. Value between 0 and 100.
  - *Gen 21:* Priority of Shield Pack. Value between 0 and 100.
  - *Gen 22:* Priority of Lightning Gun. Value between 0 and 100.
  - *Gen 23:* Priority of Shock Rifle. Value between 0 and 100.
  - *Gen 24:* Priority of Flak Cannon and of Rocket Launcher. Value between 0 and 100.
  - *Gen 25:* Priority of Minigun. Value between 0 and 100.

3.2 Fitness and evaluation

The *fitness function* is defined as:

$$f(fr, d, dmgG, dmgT, s1, s2) = \begin{cases} (fr - d) + s2 + \frac{s1}{2} \\ +log((dmgG - dmgT) + 1) \quad if \quad fr \geqslant d \\ \frac{fr}{d} + s2 + \frac{s1}{2} \\ +log((dmgG - dmgT) + 1) \quad if \quad fr < d \end{cases}$$

Where fr is the number of enemy kills the bot has obtained (frags), d is the number of own deads, $dmgG$ is the total damage produced by the bot, and $dmgT$ is the total damage it has received. $s1$ and $s2$ refers respectively to the number of Shields and Super Shields the bot has picked up (very important item). This function rewards the individuals with a positive balance (more frags than deads) and a high number of frags. In addition individuals which perform a high amount of damage to the enemies are also rewarded, even if they have not got a good balance. The logarithms are used due to the magnitude that the damages take, being around the thousand. We add 1 to avoid negative values.

Normally the fitness functions must be less complex, in this case, just considering overall the number of frags and deads. However, we have decided to consider several factors to value the performance of a bot, above all, the gathering and use of items, as they are very important in the matches in UT2K4.

The *evaluation of an individual* consists in setting the values of the chromosome in the NizorBot AI engine, then a 1 vs 1 combat is launched between this and a standard UT2K4 bot in its maximum difficulty level (they are more robust and reliable). Once the time defined for the match is finished, the summary of the individual (bot) performance regarding these values is considered for the fitness computation.

There is a high pseudo-stochastic component in these battles to take into account, since the results do not depend completely on our bot, but also on the enemy's actions which we cannot control. Thus, the fitness function is considered as *noisy* [11], since an individual could be valued as good in one combat, but yield very bad results in another match. This problem will be addressed in future works.

3.3 Selection and Genetic Operators

A *probability roulette wheel* has been used as selection mechanism, considering the fitness value as a proportion of this probability. The *elitism* has been implemented by replacing the five worst individuals of the new population by the five best of the current one.

The *uniform crossover* operator, so every gene of a descendent has the same probability of belonging to each one of the parents.

3.4 Interactive Evaluations

The interaction of the game expert has been conducted just stopping in some specific points of the evolution (in some generations). Then, a form with several questions is presented to him/her. It can be seen in Figure 1.

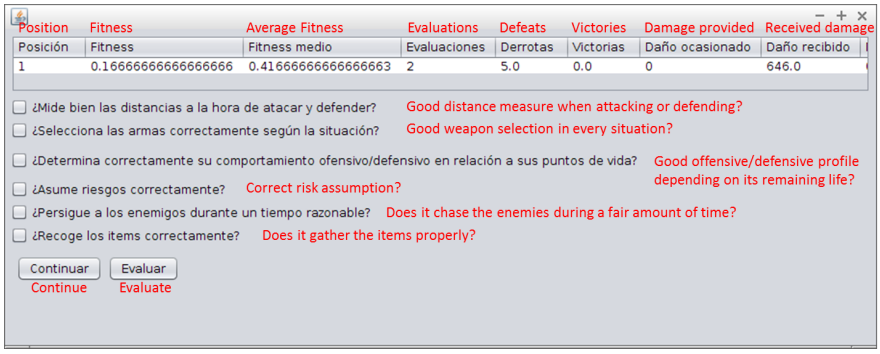

Fig. 1. Interactive evaluation form

This form shows the data about the best individual of the current generation (and thus, the best overall due to elitism). The expert can watch a video of the corresponding bot playing a match (using the number of generation and bot's ID). After the visualisation of the record, the evaluator must fill in the form, checking the boxes which better describe the behaviour he/she has seen in the video. Every checkbox is associated to a set of genes of the chromosome, so, if the expert thinks that there is a good behaviour in any sense this would be translated into the algorithm by 'freezing' (or blocking) the corresponding genes in the chromosome of the best. This will affect the rest of the population when this individual combines and spreads its genetic material. Thus, this interaction will also reduce the search space, so the algorithm performance will be improved.

The algorithm then continues the run until a new stopping point is reached.

The interaction of the algorithm expert is done by just studying the fitness evolution plotted in graphs in the software Graphic User Interface and changing the parameter values of the GA. Thus, he/she can affect the evolutionary process increasing the crossover or mutation probability, for instance.

3.5 Parallel Architecture

The evaluation of every bot/individual is very expensive in time (around 150 minutes per generation), since a match must be played almost in real time due to UT2K4 does not permit accelerating the game without provoking secondary effects, such as bad collision detections.

For this reason, we have implemented a parallel version of the algorithm, in which several individuals are evaluated at the same time (running the matches

in different machines), following a client/server architecture. We have used the extension of Pogamut [12] which let the programmer to launch matches in command line with the desired options, also including the bots with the corresponding chromosome values.

Once the match has ended, the bot (client) sends the summary/statistics to the server, in order to compute a fitness value for that bot.

If there is an error during a match, the server will be aware of this and can reinitialise the match.

4 Analysis of results

This section is devoted to analyse the performance of our proposals. To do so, we have consider the same map *DM-ironic* and a generational interactive genetic algorithm with population size 30, number of generations 50, mutation variation 10%, mutation probability 0.33%, uniform crossover and crossover probability 1.0, elitist replacement policy keeping the 5th best candidates from the previous generation. Interaction with the human expert is forced at specific generations; three versions were considered depending on the number of interactions: 1 interaction (forced at generation 25), 2 interactions (forced at generations 16 and 33), and 4 interactions (forced at generations 10,20,30 and 40). We have also considered a version with no human interaction. Five runs per algorithm, and 5 minutes for each evaluation were executed. Figure 2 shows the results obtained by these four variations of algorithm.

As interactivity increases, the results gets more consistent. Anyway, this can be the result of fixing some parts of the chromosome encoding. The consequence might be that candidates are more similar and with equivalent fitness.

4.1 Humanity evaluation

To evaluate the adequacy of our proposal, our bot competed in the 2K BotPrize competition[3] (edition 2014) that proposes the Turing test in UT2K4. This competition has been sponsored by 2K Australia. In the original competition, the evaluation of the humanity of bots was in charge of a number of judges that participated directly in the matches; this means a First Person Assessment (FPA). In the edition of 2014, a Third Person Assessment (TPA) was included by means of the participation of (external) judges via a crowdsourcing platform. The general schema of the new evaluation is shown in Figure 3 which shows the function to calculate the Humanness value (H) of a bot taking into account the FPA and TPA.

In addition, since 2014, this competition proposes a second challenge that evaluates the own skill of the bots to be able to judge the humanity of the other bots. However, this is not an objective for our bot. The results of the reliability evaluation (both by humans and also bots) is shown in Figure 4.

[3] http://botprize.org/

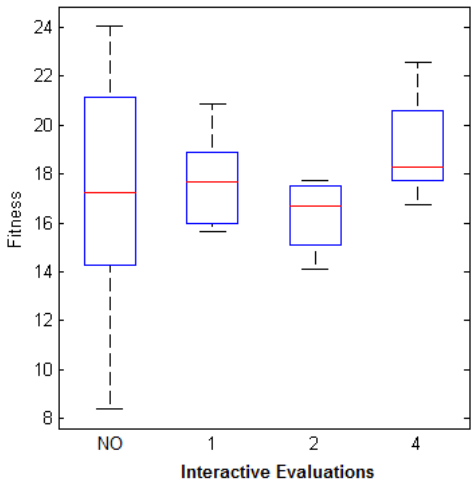

Fig. 2. Fitness comparison. In all box plots, the red line indicates the median of the distribution, whereas the bottom and top of the box correspond respectively to the first and third quartile of the distribution.

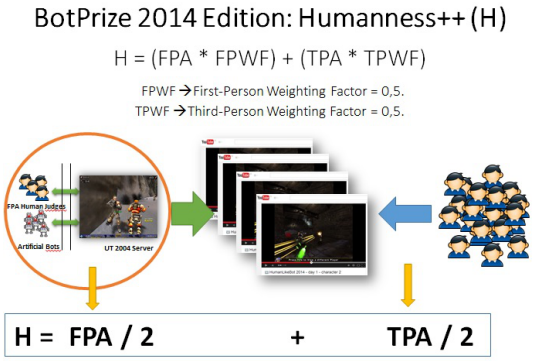

Fig. 3. Evaluation system considered in BotPrize 2014

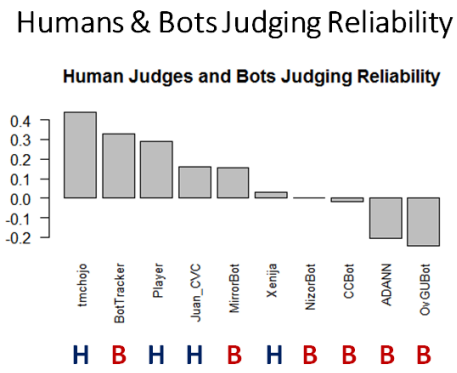

Fig. 4. Results of detection of bots (reliability) in Botprize 2014

As it can be seen in Figures 5 and 6, the winner of this edition was *Mirrorbot* (the same as in previous editions), which was one of the two bots that passed the Turing test adapted to games in the previous editions of the competition (so that it can be considered the state-of-the-art). However, as result of the new (and harder) evaluation system, it does no reach the value for being consider human (i.e., 0.5) although is relatively close to it. Out bot (NizorBot) show a very good performance finishing as the runner-up and with a humanity factor relatively close to be considered as human. it was programmed taking into account the patterns of behaviour of an expert human player in Unreal Tournament 2004, and this might explain its success.

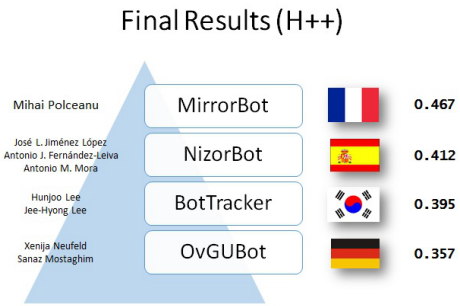

Fig. 5. Results of Botprize 2014(I)

BotName	FPA	TPA	H++
Xenija	0.17139763	0.8235294	0.4974635
MirrorBot	0.20164771	0.7333333	0.4674905
Player	0.19328127	0.6315789	0.4124301
tmchojo	0.17757519	0.6470588	0.4123170
NizorBot	0.11821633	0.7058824	0.4120493
BotTracker	0.20070203	0.5909091	0.3958056
CCBot	0.06214746	0.7058824	0.3840149
Juan_CVC	0.12372294	0.6190476	0.3713853
OvGUBot	0.10545765	0.6086957	0.3570767
ADANN	0.08351664	0.4761905	0.2798536

Fig. 6. Results of Botprize 2014(II)

5 Conclusions and future work

This paper describes a genetic algorithm-based proposal to create a Non-Player Character (bot) that plays, in a human-style, the game Unreal Tournament 2004. Some user-centric proposals (i.e., with human interaction) have been proposed and compared with a non-interactive version. All the proposals perform similarly if they are compared according only to the score (i.e., fitness) obtained during a number of matches; however, according to a humanity factor the incorporation of interactivity provides very good results, and this has been proved via the results obtained by one of our human-guided algorithms in the context of the 2K BotPrize competition.

There is room for many improvements: for instance, the schema of navigation in our bots might be improved. In addition, currently the fitness function that guides the search process introduce noise in the optimisation process, and we believe that the incorporation of several human experts in the guidance of our algorithms surely would produce more consistent results.

Acknowledgements

This work has been partially supported by project V17-2015 of the Microprojects program 2015 from CEI BioTIC Granada, Junta de Andalucía within the project P10-TIC-6083 (DNEMESIS[4]), by Ministerio de Ministerio español de Economía y Competitividad under (preselected as granted) project TIN2014-56494-C4-1-P (UMA-EPHEMECH), and Universidad de Málaga. Campus de Excelencia Internacional Andalucía Tech.

[4] http://dnemesis.lcc.uma.es/wordpress/

References

1. http://www.unrealtournament.com/: Unreal tournament (2014)
2. Arbib, M.A.: Theories of abstract automata. Prentice-Hall (1969)
3. Sims, K.: Artificial evolution for computer graphics. ACM SIGGRAPH Computer Graphics **25**(4) (1991) 319–328
4. Mora, A., Aisa, F., Caballero, R., García-Sánchez, P., Merelo, J., Castillo, P., Lara-Cabrera, R.: Designing and evolving an unreal tournamenttm 2004 expert bot. Springer (2013) 312–323
5. WEB: Gamebots project (2008) http://gamebots.sourceforge.net/.
6. Schrum, J., Karpov, I.V., Miikkulainen, R.: Ut? 2: Human-like behavior via neuroevolution of combat behavior and replay of human traces. (2011) 329–336
7. Polceanu, M.: Mirrorbot: Using human-inspired mirroring behavior to pass a turing test. (2013) 1–8
8. Klau, G., Lesh, N., Marks, J., Mitzenmacher, M.: Human-guided search. Journal of Heuristics **16** (2010) 289–310
9. Cotta, C., Leiva, A.J.F.: Bio-inspired combinatorial optimization: Notes on reactive and proactive interaction. In Cabestany, J., Rojas, I., Caparrós, G.J., eds.: Advances in Computational Intelligence - 11th International Work-Conference on Artificial Neural Networks, IWANN 2011, Part II. Volume 6692 of Lecture Notes in Computer Science., Springer (2011) 348–355
10. Badillo, A.R., Ruiz, J.J., Cotta, C., Leiva, A.J.F.: On user-centric memetic algorithms. Soft Comput. **17**(2) (2013) 285–300
11. Mora, A.M., Fernández-Ares, A., Merelo, J.J., García-Sánchez, P., Fernandes, C.M.: Effect of noisy fitness in real-time strategy games player behaviour optimisation using evolutionary algorithms. J. Comput. Sci. Technol. **27**(5) (2012) 1007–1023
12. http://pogamut.cuni.cz/main/: Pogamut - virtual characters made easy — about (2014)
13. Dyer, D.: The watchmaker framework for evolutionary computation (evolutionary/genetic algorithms for java) (2014)
14. http://human machine.unizar.es/: Human-like bots competition 2014 (2014)
15. http://xstream.codehaus.org/: Xstream - about xstream (2014)